CS 410/510: Natural Language Processing
Final Project Instructions

1 Description

Instead of a final exam, CS 410/510 for Natural Language Processing (NLP) has a final project which counts for 30% of the grade. It is intended to provide realistic experience in using or researching NLP.

There are several different ways to do this project:

- **Kaggle Competition.** Pick one of the current contests on kaggle.com, download the data, and try to develop the best model you can. Kaggle provides training data, an initial set of features, a testing framework, and a leaderboard to compare your results to many other teams. Therefore, you can focus all of your time on developing better features, expanding the training data if necessary, selecting appropriate models and tuning them, and exploring several models for performance comparision.

- **New Domain.** Identify an interesting problem, collect data, design a feature representation, apply several machine learning algorithms (being careful not to train on test data), and analyze the results. The papers on the NLP conferences (i.e., ACL, EMNLP, NAACL, COLING) will help a lot: `https://aclanthology.org/`.

- **Algorithm Development.** Develop and evaluate a new machine learning algorithm, representation, regularizer, optimization method, etc. It is hard to do this well, since most of the easy and obvious ideas have been tried already. Therefore, this kind of project is not recommended for most students.

If you want help pick a project, feel free to ask me. It's best if you already have some idea of what you want to do so we can have effective discussions. The typical problems you can consider for the final project involve: sentiment analysis, text classification, named entity recognition, part of speech tagging, dependency parsing, question answering, machine translation, and summarization. For each of these problems, you can consider different types of models, e.g., rule-based models, traditional feature-based models with different types of features, deep learning models with different types of word embeddings, or advanced pre-trained transformer-based architectures.

2 Methods and Results

Your project must contain theoretical or empirical results. Coming up with new theoretical results of interest is difficult, so I expect that most of you will only present empirical results.

For an application paper, you should evaluate and justify the choices you made. Here are some questions to think about:

- Which problems do you want to solve? Where do you obtain the data? How much data do you have? What cleaning or processing did you do to the data? (For some problems, you may need to be creative about integrating data from multiple sources, or making use of noisy labels.)
- How did you formulate your problem? Is this problem best posed as classification? Regression? Sequence Labeling? Sequence-to-Sequence?
- What features (if applicable) do you select and why?
- What algorithms/models do you want to explore? (You should use more than one, in order to have a comparison.)
- What baselines do you use for comparison (if proposing a novel algorithm or feature set or problem formulation)?
- How do you set up the training/tuning/testing data? Do you do cross-validation? How do you tune the hyper-parameters?
- How do you choose to measure performance? Accuracy? Learning curves? ROC curves? Confusion matrix? Precision/recall/F1 measure? Running time?
- Which algorithm performs best? Can you determine why that algorithm works best? It's OK to explain your hypothesis based on your observations of data and empirical results.

You do not need to implement everything yourself. scikit-learn, OpenNLP, and pytorch are popular open-source toolkits that include many common models. They also provide efficient frameworks to implement new models.

Please do follow the scientific method. Develop appropriate experiments to validate or refute your hypotheses, as well as to provide more insight. For example, which feature representation worked best? Which classifiers or combinations of classifiers worked best? Why do you think this is? What evidence do you have for this explanation?

An accuracy with no explanation is not interesting. An explanation of how you obtained that accuracy, what worked and what didn't, and what you learned is more interesting. Please include some quantitative measures, in tables, charts, and graphs.

This does not need to be publishable research, but it should demonstrate that you understand how to apply NLP techniques to a real problem (for an application paper) or how to develop and evaluate novel algorithms (for an algorithms paper).

Negative results are acceptable. If you get a negative result, explore what happened and why. Not enough data? Overfitting? Bad features? Noisy labels?

Different distribution at test time? Explore what led to the poor results and try to determine if that could be overcome.

You can also try ambitious projects that might be hard to complete during this course, but please talk to the instructor about this in advance. Ambitious projects will be evaluated based on the status of the projects at the time of the final reports. A detailed plan for the project, where you are at the submission time, and how you plan to finish it should be presented in the final reports.

3 Writing

All papers are expected to be clearly written with a good structure. I will hold graduate students to a higher standard of formal, technical writing and analysis of experimental results.

Many NLP papers use a structure similar to the following (see `https://www.aclweb.org/anthology/` for many examples):

1. Abstract: Summarize the entire paper (including results) in 50-250 words.

2. Introduction: Identify the problem you're trying to solve, describe why it's important, and outline the key method or strategy that you will use to solve it.

3. Related work: Describe the previous work that are related to your current work. This can be done in different aspects, i.e., models, features, datasets, applications etc. A good related work section shows your background and capacity to work on this problem.

4. Methods: Describe the technologies or ideas that you will build on in your method. For an application paper, this could simply be a detailed description of the problem you're trying to solve and the methods you're applying. For an algorithm paper, this could be the NLP/machine learning methods that you're extending. If you present a new method, clearly describe the motivation and the actual implementation.

   should contain your key contributions.

5. Experiments: Evaluate your approach experimentally. Describe your methods in enough detail that another researcher could replicate them (i.e., what are the hyper-parameters? how do you select them? What are the values you've used?). How well does your method work? Does your method outperform reasonable baselines? How does your method compare to simplified versions of your method? What kinds of errors remain? What interesting things do you learn from your experiments? Tables of results are useful, but charts and figures are often better. This can also be integrated with the methods, so that each aspect of the model is evaluated as it is introduced (e.g., feature selection, classifier selection, ensemble construction).

6. Conclusion: Summarize your contributions and discuss future work (50-500 words).

7. References: Works that you cite in the body of your paper. You may use any standard citation style as long as it is consistent.

I recommend that you use a structure similar to this one, unless you have a good reason for another structure.

I do not require perfect English, but I greatly appreciate clear writing. Your methods should be described clearly enough to replicate your results. Your conclusions should be supported by evidence. Your arguments should follow logically. Each paragraph should discuss a single idea. If you're having trouble, there is writing tutoring available on campus for all students.

Learning to write a good technical paper is an extremely valuable skill in both graduate school and industry. Writing well is very difficult, even for experienced writers, but it does get easier with practice.

Students are encouraged to use the NAACL template to write the report for this class: `http://naacl.org/naacl-pubs`.

4 Grading

The final project will count for 30% of the overall score. 5% will be given to on time submission for final project proposal while the other 25% will be based on project results, reports and code. As such, we will give you 16 points if you submit the final project proposal on time. We will reserve 80 points for project results, reports, and code (so the maximal is 96 points). The break-down scores are shown below:

- On time proposal for the final project - 16 points

- Paper, Code - 30 points. Paper should be clearly written and structured. Use tables, figures, and other visualizations as appropriate. Description of the problems and the methods and presentation of results should showcase the scientific method – make it clear what you're evaluating, how you're evaluating it, and what the result is. Discussion and analysis of the results. The code should be well structured and include a README file to show how to run the code.

- Methods - 50 points. Select appropriate feature representations and models (10 points). Evaluate a variety of methods, select methods that are a good fit to the particular problem you're working on (10 points). Use proper experimental procedures for parameter tuning and model evaluation (14 points). Explore some additional features/models that works on your problem (16 points).